MSCA32013 Optimization & Simulation Methods for Analytics

Final Project - Kyoto Restaurants

By: WanQi(Angie) Tay, Alisa Babikova, Yingkun(Will) Zhu

In [1]:
```
import pandas as pd
kyoto = pd.read_csv("Kyoto.csv")
kyoto.head()
```

Out[1]:

	Column	X	Column2	Name	JapaneseName	Station	FirstCategory	SecondCategory	DinnerPrice	Di
0	1	1	1	Orudeidainingurajou	オールデイダイニング　ラジョウ	Kyoto	Buffet style	Cafe	￥4000～￥4999	40
1	2	2	2	Steak Frites Gaspard zinzin	ステックフリット ガスパール ザンザン	Karasuma	Bistro	Steak	￥3000～￥3999	30
2	3	3	3	KAZUMA	和馬	Sanjo	Izakaya (Tavern)	Japanese food (other)	￥3000～￥3999	30
3	4	4	4	okonomiyakiteppanyakimiki	お好み焼き 鉄板焼き 三喜	Tambaguchi	Okonomiyaki	Izakaya (Tavern)	￥3000～￥3999	30
4	8	8	8	Sumika	棲家	Kawaramachi	Izakaya (Tavern)	Creative cuisine	￥3000～￥3999	30

5 rows × 26 columns

In [2]:
```
new_kyoto = kyoto[['Name','AverageDinnerPriceInUSD','AverageLunchPriceInUSD','DinnerRating','LunchRating','Categor
y']]
new_kyoto.head(5)
```

Out[2]:

	Name	AverageDinnerPriceInUSD	AverageLunchPriceInUSD	DinnerRating	LunchRating	Category
0	Orudeidainingurajou	40.452161	22.469427	3.20	3.38	Buffet
1	Steak Frites Gaspard zinzin	31.460794	13.478060	3.06	3.33	Bistro
2	KAZUMA	31.460794	0.000000	3.28	0.00	Other
3	okonomiyakiteppanyakimiki	31.460794	0.000000	3.14	0.00	Okonomiyaki
4	Sumika	31.460794	0.000000	3.34	0.00	Fusion

In [4]:
```
# Convert all NaN's and NA's to 0

import warnings
warnings.filterwarnings("ignore")
from numpy import *
new_kyoto.fillna(0)
where_are_NaNs = isnan(new_kyoto['LunchRating'])
new_kyoto['LunchRating'][where_are_NaNs] = 0
new_kyoto.head(5)
```

Out[4]:

	Name	AverageDinnerPriceInUSD	AverageLunchPriceInUSD	DinnerRating	LunchRating	Category
0	Orudeidainingurajou	40.452161	22.469427	3.20	3.38	Buffet
1	Steak Frites Gaspard zinzin	31.460794	13.478060	3.06	3.33	Bistro
2	KAZUMA	31.460794	0.000000	3.28	0.00	Other
3	okonomiyakiteppanyakimiki	31.460794	0.000000	3.14	0.00	Okonomiyaki
4	Sumika	31.460794	0.000000	3.34	0.00	Fusion

In [6]:
```python
# Dinner Model
from pymprog import *
import numpy as np


preference = ("Buffet", "Bistro", "Fusion", "Bar") # haven't figured this out yet
days = 2
budget = [50, 100]
df = new_kyoto[new_kyoto["Category"].isin(preference)]
n = len(df)
A= list(new_kyoto['DinnerRating'])
c = list(new_kyoto['AverageDinnerPriceInUSD'])

begin('dinner') # begin modelling
verbose(True)
x = var('x',n, kind = bool) # set up the variiable
rating = sum(A[i]*x[i] for i in range(n))/days # initialize objective function
maximize(rating) # define the direction
#  Set constraints
R1 = sum(p*q for p,q in zip(c,x)) >= min(budget)
R2 = sum(p*q for p,q in zip(c,x)) <= max(budget)
R3 = sum(x[i] for i in range(n)) == days

solve()
end
```

```
Max : ( 3.2 * x[0] + 3.06 * x[1] + 3.28 * x[2] + 3.14 * x[3] + 3.34 * x[4] + 3.04 * x[5] + 3.56 * x[6] + 3.04 * x
[7] + 3.0 * x[8] + 3.05 * x[9] + 3.1 * x[10] + 3.1 * x[11] + 3.06 * x[12] + 3.07 * x[13] + 3.03 * x[14] + 3.66 * x
[15] + 3.09 * x[16] + 3.03 * x[17] + 3.06 * x[18] + 3.35 * x[19] + 3.47 * x[20] + 3.17 * x[21] + 3.06 * x[22] + 3.
02 * x[23] + 3.03 * x[24] + 3.06 * x[25] + 3.25 * x[26] + 3.53 * x[27] + 3.04 * x[28] + 3.05 * x[29] + 3.17 * x[3
0] + 3.45 * x[31] + 3.08 * x[32] + 3.06 * x[33] + 3.02 * x[34] + 3.09 * x[35] + 3.51 * x[36] + 3.47 * x[37] + 3.12
* x[38] + 3.04 * x[39] + 3.05 * x[40] + 3.42 * x[41] + 3.33 * x[42] + 3.13 * x[43] + 3.03 * x[44] + 3.08 * x[45] +
3.54 * x[46] + 3.03 * x[47] + 3.54 * x[48] + 3.04 * x[49] + 3.05 * x[50] + 3.08 * x[51] + 3.64 * x[52] + 3.01 * x
[53] + 3.51 * x[54] + 3.32 * x[55] + 3.14 * x[56] + 3.5 * x[57] ) / 2
R1: 40.45216148 * x[0] + 31.46079418 * x[1] + 31.46079418 * x[2] + 31.46079418 * x[3] + 31.46079418 * x[4] + 31.46
079418 * x[5] + 40.45216148 * x[6] + 31.46079418 * x[7] + 22.46942688 * x[8] + 49.44352877 * x[9] + 31.46079418 *
x[10] + 22.46942688 * x[11] + 31.46079418 * x[12] + 49.44352877 * x[13] + 31.46079418 * x[14] + 80.91331432 * x[1
5] + 40.45216148 * x[16] + 31.46079418 * x[17] + 31.46079418 * x[18] + 31.46079418 * x[19] + 49.44352877 * x[20] +
22.46942688 * x[21] + 49.44352877 * x[22] + 31.46079418 * x[23] + 31.46079418 * x[24] + 49.44352877 * x[25] + 40.4
5216148 * x[26] + 31.46079418 * x[27] + 31.46079418 * x[28] + 40.45216148 * x[29] + 40.45216148 * x[30] + 112.3830
999 * x[31] + 40.45216148 * x[32] + 49.44352877 * x[33] + 31.46079418 * x[34] + 31.46079418 * x[35] + 31.46079418
* x[36] + 112.3830999 * x[37] + 22.46942688 * x[38] + 31.46079418 * x[39] + 40.45216148 * x[40] + 31.46079418 * x
[41] + 31.46079418 * x[42] + 31.46079418 * x[43] + 49.44352877 * x[44] + 40.45216148 * x[45] + 112.3830999 * x[46]
+ 40.45216148 * x[47] + 112.3830999 * x[48] + 40.45216148 * x[49] + 31.46079418 * x[50] + 31.46079418 * x[51] + 11
2.3830999 * x[52] + 31.46079418 * x[53] + 22.46942688 * x[54] + 112.3830999 * x[55] + 22.46942688 * x[56] + 31.460
79418 * x[57] >= 50
R2: 40.45216148 * x[0] + 31.46079418 * x[1] + 31.46079418 * x[2] + 31.46079418 * x[3] + 31.46079418 * x[4] + 31.46
079418 * x[5] + 40.45216148 * x[6] + 31.46079418 * x[7] + 22.46942688 * x[8] + 49.44352877 * x[9] + 31.46079418 *
x[10] + 22.46942688 * x[11] + 31.46079418 * x[12] + 49.44352877 * x[13] + 31.46079418 * x[14] + 80.91331432 * x[1
5] + 40.45216148 * x[16] + 31.46079418 * x[17] + 31.46079418 * x[18] + 31.46079418 * x[19] + 49.44352877 * x[20] +
22.46942688 * x[21] + 49.44352877 * x[22] + 31.46079418 * x[23] + 31.46079418 * x[24] + 49.44352877 * x[25] + 40.4
5216148 * x[26] + 31.46079418 * x[27] + 31.46079418 * x[28] + 40.45216148 * x[29] + 40.45216148 * x[30] + 112.3830
999 * x[31] + 40.45216148 * x[32] + 49.44352877 * x[33] + 31.46079418 * x[34] + 31.46079418 * x[35] + 31.46079418
* x[36] + 112.3830999 * x[37] + 22.46942688 * x[38] + 31.46079418 * x[39] + 40.45216148 * x[40] + 31.46079418 * x
[41] + 31.46079418 * x[42] + 31.46079418 * x[43] + 49.44352877 * x[44] + 40.45216148 * x[45] + 112.3830999 * x[46]
+ 40.45216148 * x[47] + 112.3830999 * x[48] + 40.45216148 * x[49] + 31.46079418 * x[50] + 31.46079418 * x[51] + 11
2.3830999 * x[52] + 31.46079418 * x[53] + 22.46942688 * x[54] + 112.3830999 * x[55] + 22.46942688 * x[56] + 31.460
79418 * x[57] <= 100
R3: (x[0] + x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7] + x[8] + x[9] + x[10] + x[11] + x[12] + x[13] + x[14] +
x[15] + x[16] + x[17] + x[18] + x[19] + x[20] + x[21] + x[22] + x[23] + x[24] + x[25] + x[26] + x[27] + x[28] + x
[29] + x[30] + x[31] + x[32] + x[33] + x[34] + x[35] + x[36] + x[37] + x[38] + x[39] + x[40] + x[41] + x[42] + x[4
3] + x[44] + x[45] + x[46] + x[47] + x[48] + x[49] + x[50] + x[51] + x[52] + x[53] + x[54] + x[55] + x[56] + x[57]
==2)
__del__ is deleting problem: dinner
```

Out[6]:
```
<function pymprog.end>
```

MIP outputs 3.545 as integer optimal solution and 3.582 as linear optimal solution

In [7]:
```
# Lunch Model
from pymprog import *
import numpy as np

preference = ("Buffet", "Bistro", "Fusion", "Bar", "Yakitori")
days = 2
budget = [70, 100]
df = new_kyoto[(new_kyoto['Category'] == 'Buffet') | (new_kyoto['Category'] == "Bistro") |
          (new_kyoto['Category'] == 'Fusion') | (new_kyoto['Category'] == "Bar")]
n = len(df)
A = list(new_kyoto['LunchRating'])
c = list(new_kyoto['AverageLunchPriceInUSD'])

begin('lunch') # begin modelling
verbose(True)
x = var('x',n, kind = bool) # set up the variiable
rating = sum(A[i]*x[i] for i in range(n))/days # initialize objective function
maximize(rating) # define the direction
# set constraints
R1 = sum(p*q for p,q in zip(c,x)) >= min(budget)
R2 = sum(p*q for p,q in zip(c,x)) <= max(budget)
R3 = sum(x[i] for i in range(n)) == days
solve()
end()
```

```
Max : ( 3.38 * x[0] + 3.33 * x[1] + 0.0 * x[2] + 0.0 * x[3] + 0.0 * x[4] + 0.0 * x[5] + 0.0 * x[6] + 0.0 * x[7] +
0.0 * x[8] + 0.0 * x[9] + 0.0 * x[10] + 3.06 * x[11] + 0.0 * x[12] + 3.55 * x[13] + 0.0 * x[14] + 3.58 * x[15] +
0.0 * x[16] + 3.02 * x[17] + 0.0 * x[18] + 3.04 * x[19] + 3.16 * x[20] + 0.0 * x[21] + 0.0 * x[22] + 0.0 * x[23] +
0.0 * x[24] + 0.0 * x[25] + 0.0 * x[26] + 3.32 * x[27] + 0.0 * x[28] + 0.0 * x[29] + 3.03 * x[30] + 3.59 * x[31] +
0.0 * x[32] + 0.0 * x[33] + 0.0 * x[34] + 0.0 * x[35] + 0.0 * x[36] + 0.0 * x[37] + 0.0 * x[38] + 0.0 * x[39] + 0.
0 * x[40] + 0.0 * x[41] + 3.35 * x[42] + 3.1 * x[43] + 3.03 * x[44] + 3.52 * x[45] + 3.53 * x[46] + 0.0 * x[47] +
3.58 * x[48] + 0.0 * x[49] + 0.0 * x[50] + 3.06 * x[51] + 3.68 * x[52] + 3.0 * x[53] + 3.56 * x[54] + 3.45 * x[55]
+ 0.0 * x[56] + 3.32 * x[57] ) / 2
R1: 22.46942688 * x[0] + 13.47805958 * x[1] + 0.0 * x[2] + 0.0 * x[3] + 0.0 * x[4] + 0.0 * x[5] + 0.0 * x[6] + 0.0
* x[7] + 22.46942688 * x[8] + 4.486692282 * x[9] + 0.0 * x[10] + 4.486692282 * x[11] + 13.47805958 * x[12] + 22.46
942688 * x[13] + 0.0 * x[14] + 22.46942688 * x[15] + 0.0 * x[16] + 4.486692282 * x[17] + 0.0 * x[18] + 0.0 * x[19]
+ 4.486692282 * x[20] + 0.0 * x[21] + 0.0 * x[22] + 0.0 * x[23] + 13.47805958 * x[24] + 0.0 * x[25] + 0.0 * x[26]
+ 22.46942688 * x[27] + 0.0 * x[28] + 0.0 * x[29] + 13.47805958 * x[30] + 49.44352877 * x[31] + 0.0 * x[32] + 4.48
6692282 * x[33] + 0.0 * x[34] + 0.0 * x[35] + 0.0 * x[36] + 0.0 * x[37] + 0.0 * x[38] + 4.486692282 * x[39] + 0.0
* x[40] + 0.0 * x[41] + 13.47805958 * x[42] + 4.486692282 * x[43] + 0.0 * x[44] + 22.46942688 * x[45] + 31.4607941
8 * x[46] + 0.0 * x[47] + 40.45216148 * x[48] + 4.486692282 * x[49] + 0.0 * x[50] + 0.0 * x[51] + 49.44352877 * x
[52] + 0.0 * x[53] + 4.486692282 * x[54] + 40.45216148 * x[55] + 0.0 * x[56] + 4.486692282 * x[57] >= 70
R2: 22.46942688 * x[0] + 13.47805958 * x[1] + 0.0 * x[2] + 0.0 * x[3] + 0.0 * x[4] + 0.0 * x[5] + 0.0 * x[6] + 0.0
* x[7] + 22.46942688 * x[8] + 4.486692282 * x[9] + 0.0 * x[10] + 4.486692282 * x[11] + 13.47805958 * x[12] + 22.46
942688 * x[13] + 0.0 * x[14] + 22.46942688 * x[15] + 0.0 * x[16] + 4.486692282 * x[17] + 0.0 * x[18] + 0.0 * x[19]
+ 4.486692282 * x[20] + 0.0 * x[21] + 0.0 * x[22] + 0.0 * x[23] + 13.47805958 * x[24] + 0.0 * x[25] + 0.0 * x[26]
+ 22.46942688 * x[27] + 0.0 * x[28] + 0.0 * x[29] + 13.47805958 * x[30] + 49.44352877 * x[31] + 0.0 * x[32] + 4.48
6692282 * x[33] + 0.0 * x[34] + 0.0 * x[35] + 0.0 * x[36] + 0.0 * x[37] + 0.0 * x[38] + 4.486692282 * x[39] + 0.0
* x[40] + 0.0 * x[41] + 13.47805958 * x[42] + 4.486692282 * x[43] + 0.0 * x[44] + 22.46942688 * x[45] + 31.4607941
8 * x[46] + 0.0 * x[47] + 40.45216148 * x[48] + 4.486692282 * x[49] + 0.0 * x[50] + 0.0 * x[51] + 49.44352877 * x
[52] + 0.0 * x[53] + 4.486692282 * x[54] + 40.45216148 * x[55] + 0.0 * x[56] + 4.486692282 * x[57] <= 100
R3: (x[0] + x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7] + x[8] + x[9] + x[10] + x[11] + x[12] + x[13] + x[14] +
x[15] + x[16] + x[17] + x[18] + x[19] + x[20] + x[21] + x[22] + x[23] + x[24] + x[25] + x[26] + x[27] + x[28] + x
[29] + x[30] + x[31] + x[32] + x[33] + x[34] + x[35] + x[36] + x[37] + x[38] + x[39] + x[40] + x[41] + x[42] + x[4
3] + x[44] + x[45] + x[46] + x[47] + x[48] + x[49] + x[50] + x[51] + x[52] + x[53] + x[54] + x[55] + x[56] + x[57]
==2)
__del__ is deleting problem: dinner
```

Out[7]:
```
model('lunch') is not the default model.
```

MIP identifies 3.635 as both linear and integer optimal solutions